Approximating Noisy Nonlinear Oscillator Dynamics using Markov Chains

CHRISTOPHER MOAKLER

Nonlinear Oscillators

- Nonlinear oscillators can take many forms but one of the most commonly studied is that of the Duffing Oscillator.
- The Duffing Oscillator is described by:

$$\ddot{x} + \delta\dot{x} + \alpha x + \beta x^3 = \gamma\cos(\omega t)$$

- The various parameters are
  - x: displacement of the oscillator
  - δ: controls the degree of damping
  - α: controls the stiffness
  - β: controls the degree of non-linearity
  - γ: controls the amplitude of the driving force
  - ω: the angular frequency of the driving force

Duffing Equation with noise

- The Duffing equation models the behavior of a mass attached to a non-linear spring and linear dampener.
- The system is also subject to a periodic driving force.
- We are interested in studying the Duffing equation with an added noise term.
- That is,

$$\ddot{x} + \delta\dot{x} + \alpha x + \beta x^3 = \gamma\cos(\omega t) + \sqrt{\epsilon}\, dW$$

- In general, we can not find exact solutions to the Duffing equation and instead we turn to numerical techniques.

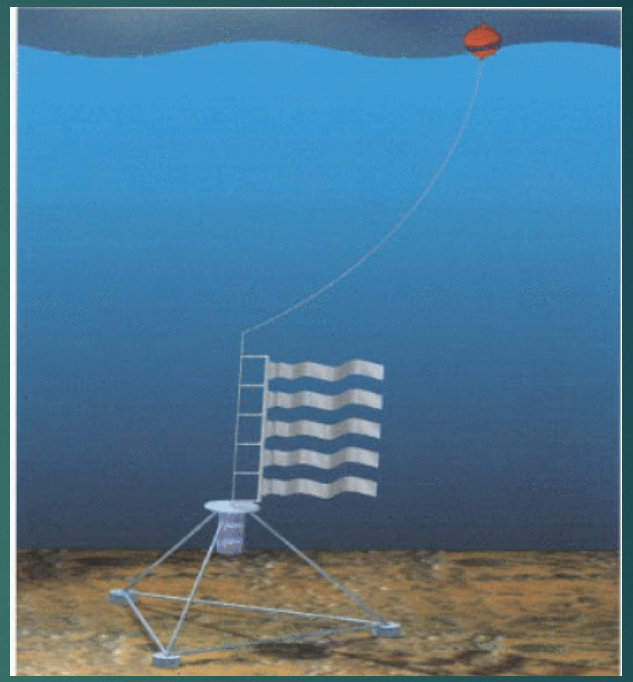
Motivations

- What makes these systems interesting?
- The Duffing equation can admit multiple attractors which are stable periodic orbits in the state space.
- Some interesting work done by a recent PhD from UMD, Lautaro Cilenti, explored these attractors and their applications to systems such as turbines and vibrational energy harvesters.

Motivating Examples

https://blog.klm.com/jet-engine-propulsion-the-comparison-of-power-between-a-car-and-an-aircraft/

Turbines have circular arrays of blades that can be modeled as non-linear oscillators

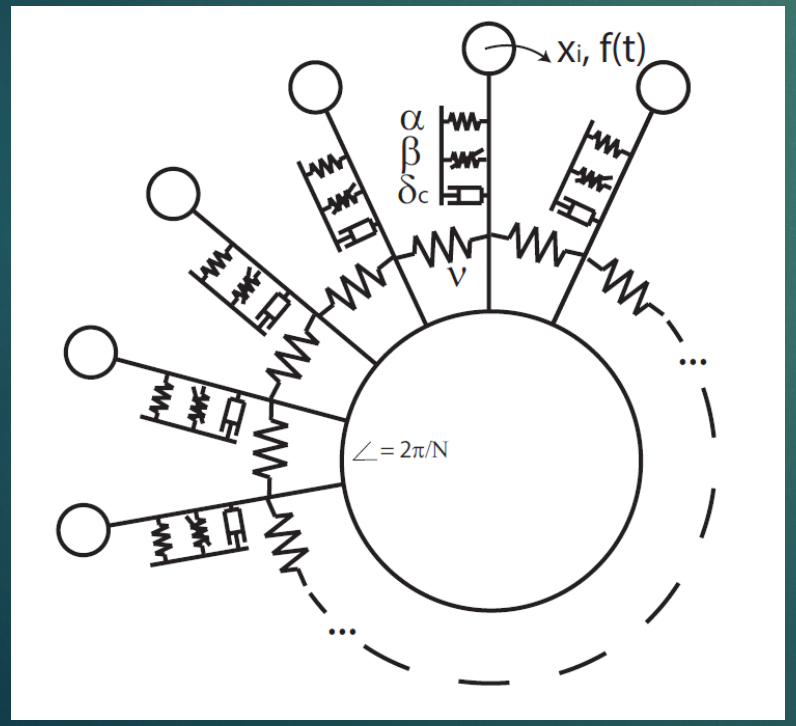


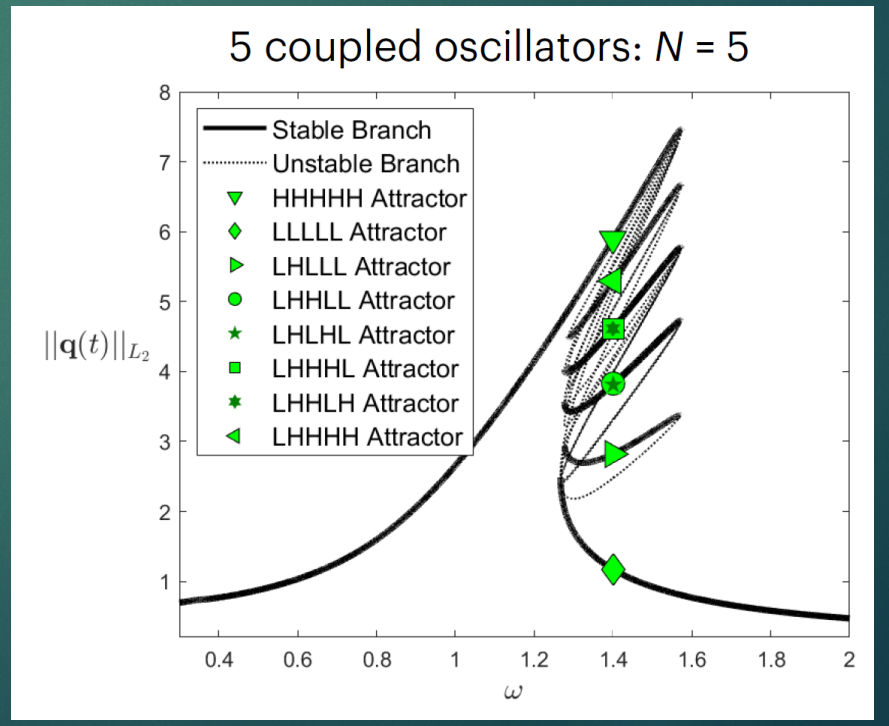
Taylor et al *The Energy Harvesting Eel: a small subsurface ocean/river power generator* 2001

Vibrational energy can be harvested in myriad ways.

Controlling the Vibrations with Noise

- Cilenti did very interesting work looking at coupled non-linear oscillators in a circular array.
- These oscillators are coupled through the base that connects them.





Cilenti 2022

Controlling the Vibrations with Noise

- The most probable escape paths between oscillators were found using a combination of large deviation theory, optimal control theory, and Floquet theory.
- Cilenti was successful in determining the most probable escape paths and quasipotential barriers using this approach for single oscillators as well as 2, 3, and 5 coupled oscillators.
- He was unable to determine the escape rate which is something we're interested in finding.

Analogue Markov Chain Approach

- The approach we're taking is to model the process with an analogue Markov chain proposed originally by Lorenz.
- The analogue Markov chain approach discretizes the phase space into sets of distinct states.
- The transition probability between these states is then approximated.
- With this transition probability matrix, we can calculate the committor function.
- A proof-of-concept for this approach was undertaken by Daniel Yuan during the REU last year at UMD.

Brief Review of Markov Chains

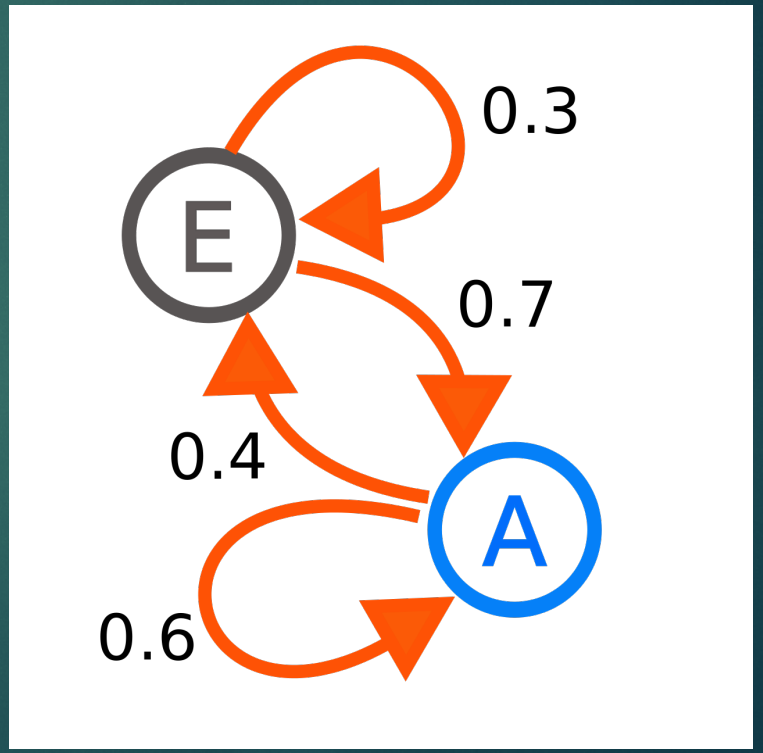
- Professor Cameron went over Markov Chains with you previously, but I'll do a quick review.
- First, what is a Markov Chain or Markov Process?
- A Markov Process is a sequence of events where the probability of each event depends only on the current state.
- What is an example of a Markov Process?
- Flipping a coin, rolling a dice, roulette, etc.
- These processes don't have "memory"
- What are examples that are NOT Markovian?

Random Walk

- A common example of a Markov Process is that of a "random walk".
- At each time step, you randomly choose a direction to take your next step in.
- When I was taught this, we called it a drunkard's walk and answered questions like, "Will the drunk find it home from the bar?"
- The drunk takes each step randomly without remembering where they were.
- You can study this in 1-, 2-, 3-, or even more dimensions if you like!

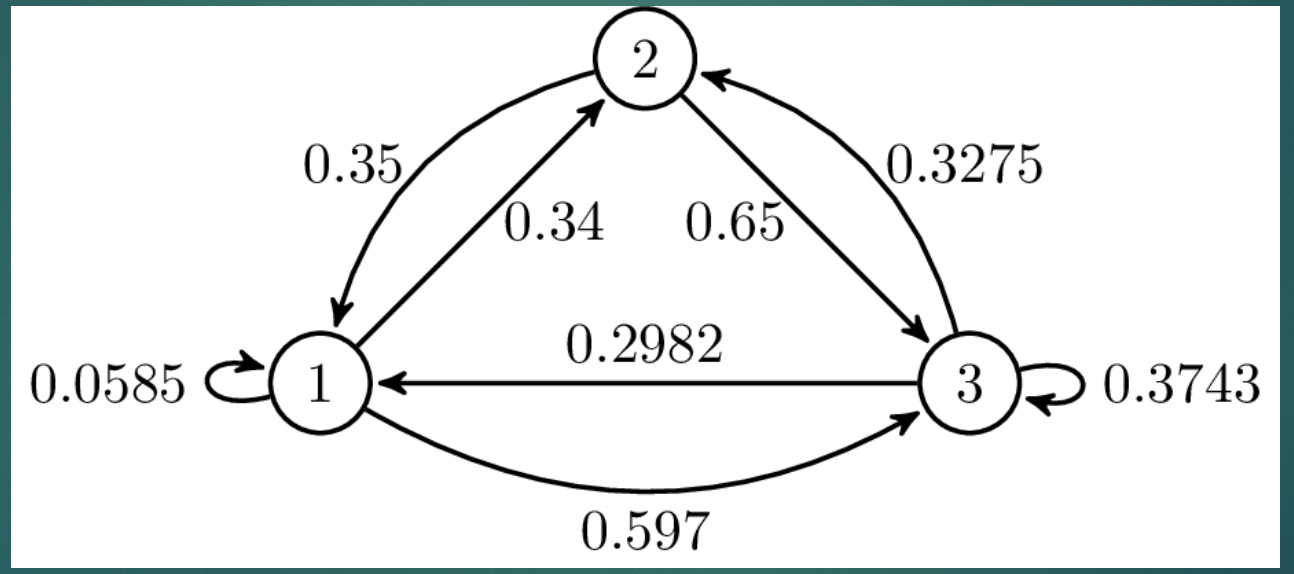
Markov Transition Probability Matrix

- It can be convenient to represent the probabilities of going from one state to another with a matrix.
- Let's construct one for this process.
- How many states are there?
- How large of a matrix will we need?
- What are the values of each matrix element.



A More Complex Example

- Let's try it for this example,



How to determine the Markov Matrix

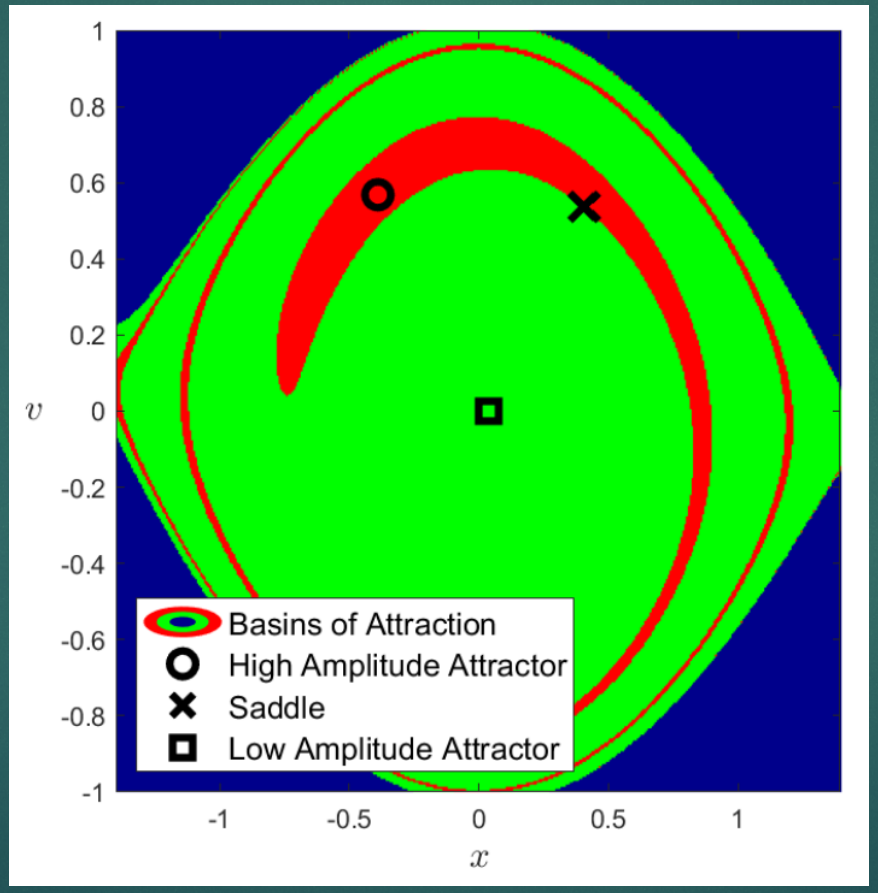
- To construct the Markov Transition Matrix, we follow a series of steps

1. Generate the Point Cloud
   - We generate a point cloud that samples the transition the pathway and the are around the attractors
2. From the point cloud, we simulate trajectories for one period.
   - We launch a large number of trajectories and track where they begin and end.
3. We identify which states in the point cloud are closest to the end of the trajectories and consider them to have ended there instead.
4. We form a transition probability matrix from this data.

Generating the Point Cloud

- To generate the point cloud, we run two very long trajectories (~1000+ periods) originating from the two attractors.
- You can run these trajectories at a given noise value to sample the space that the system is likely to encounter.
- However, you get pretty awful point clouds in this way.

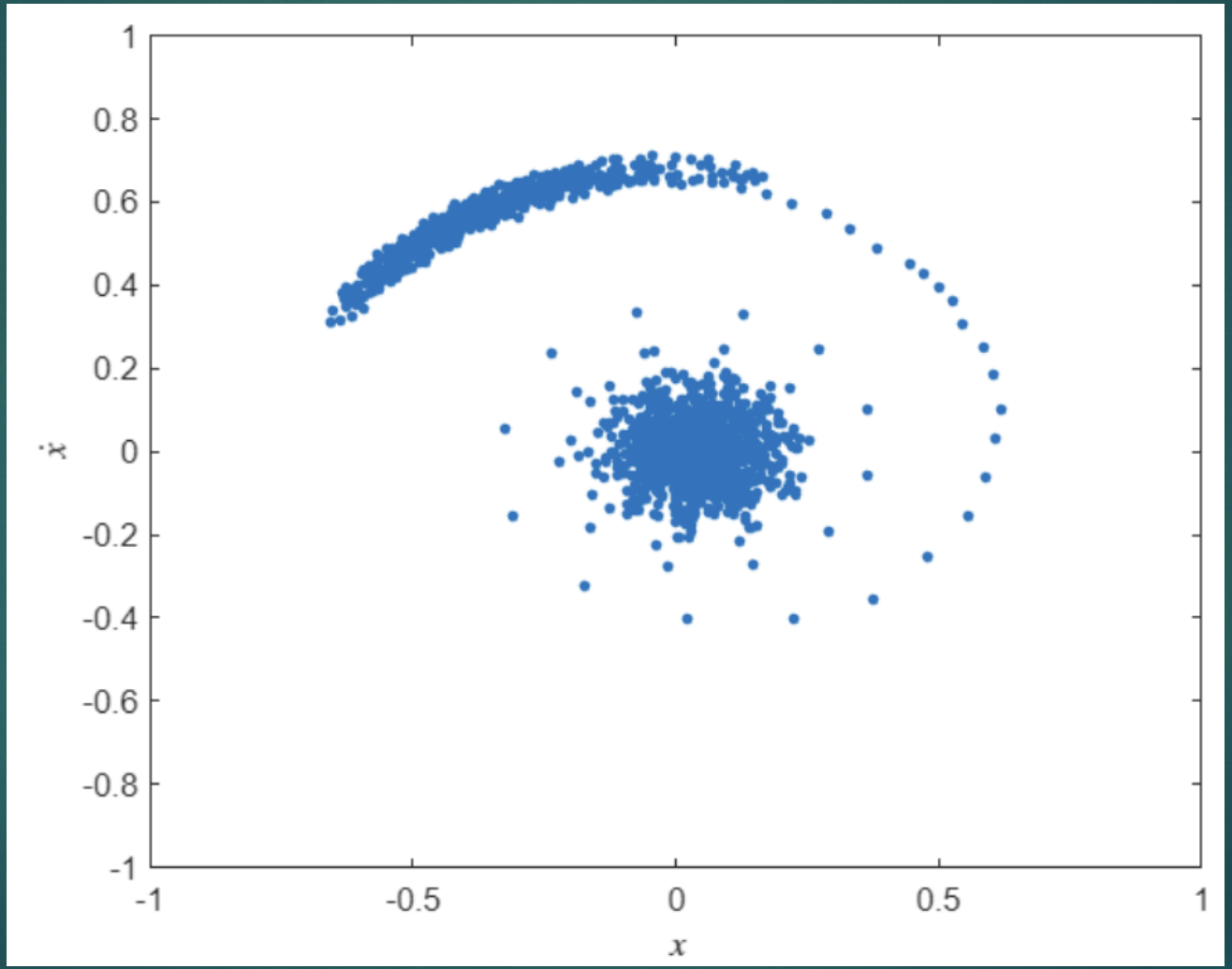
The Basins from Cilenti's Work



Lautaro 2022

The Blue region is an unstable basin that explodes to infinity.

A Bad Point Cloud



Basic Sampling

- That "basic sampling" does a very poor job of sampling the space between the basins.
- There is the high amplitude attractor with a relatively small basin.
- The trajectory quickly leaves that basin and moves into the low amplitude attractor's basin.
- However, the path that the system takes in that region is where a lot of the important dynamics are taking place!
- We want to sample that better.
- How can we do that?
- What does it even mean to "sample better"?

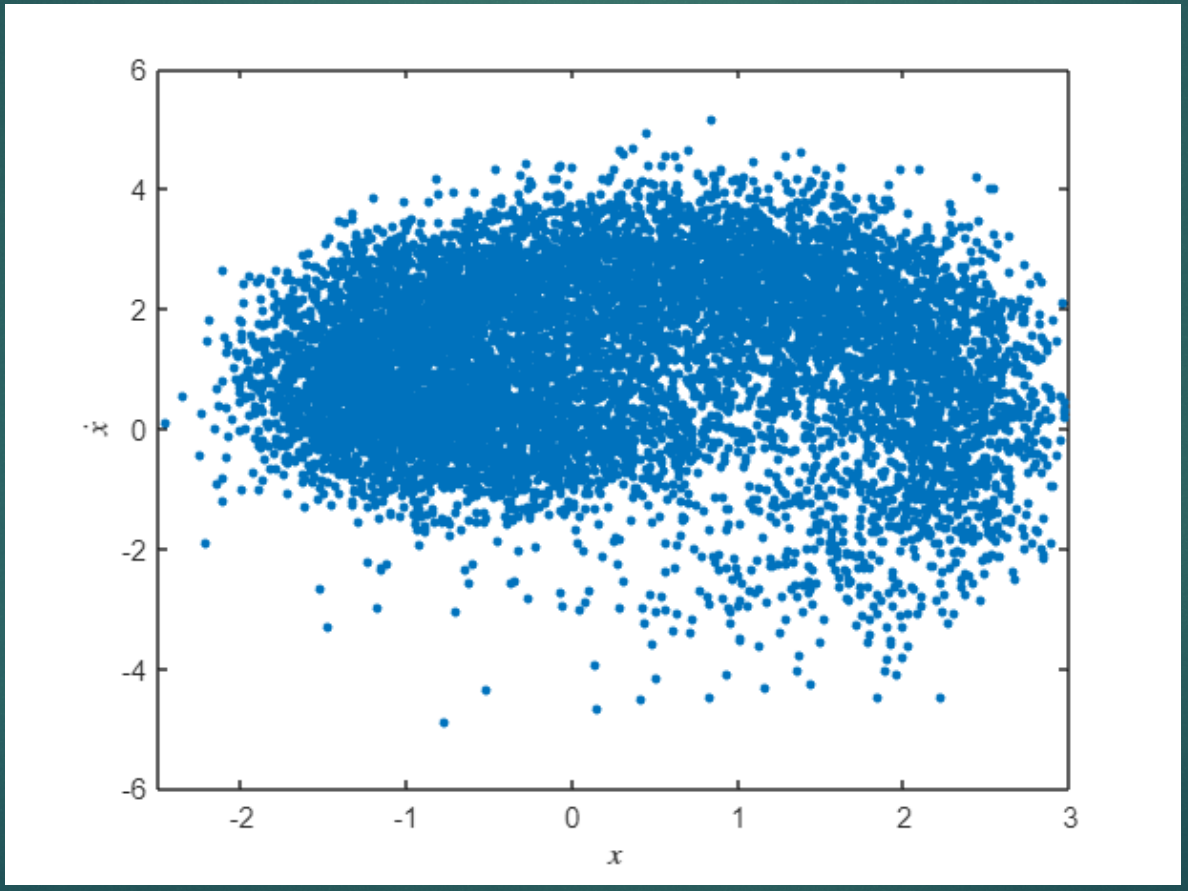
Enhanced Sampling

- In the study of rare events, getting a good sample is essentially the whole ball game.
- Getting a good sample is… nontrivial.
- Since rare events are so rare, it can take a long time to see them occur.
- Typically, we want to see them occur frequently so we can get a large number of samples.
- There are a whole host of "enhanced sampling" methods.
- Some of these simply "raise the temperature" of the simulation.
- There are a lot of different methods that I won't go into.

Metadynamics

- Metadynamics is one of the more popular enhanced sampling approaches.
- In metadynamics, you keep track of where the system has been during the trajectory and periodically add a bias to push it away from where it's been.
- For example, if the trajectory has spent the past 100 time steps around the origin, we add a bump there to push the system away.
- After another 100 timesteps, maybe the system is around (1,0), so we add another bump there.
- Over a long time, we add many bumps and sample the space in harder to reach areas.
- I think about it as adding dirt to the Grand Canyon.
  - Add enough little dirt piles and you can just walk out of the Grand Canyon.
  - The park rangers will not like this though…

Point cloud



Spatially Rarefy the States

- That picture on the previous point is a bit of a lie.
- When you run the metadynamics sampling, you get a LOT of states.
- It's basically a solid blob.
- That's a bit too many points for what we need so we "rarefy" the states.
  - Rarefy as in make more rare.
- The way we do this is by removing points that are close to other points.
  - There's more to it than just this but I'll come back to it.

How to determine the Markov Matrix

- To construct the Markov Transition Matrix, we follow a series of steps

1. Generate the Point Cloud
   - We generate a point cloud that samples the transition the pathway and the are around the attractors
2. From the point cloud, we simulate trajectories for one period.
   - We launch a large number of trajectories and track where they begin and end.
3. We identify which states in the point cloud are closest to the end of the trajectories and consider them to have ended there instead.
4. We form a transition probability matrix from this data.

Trajectories from the Point Cloud

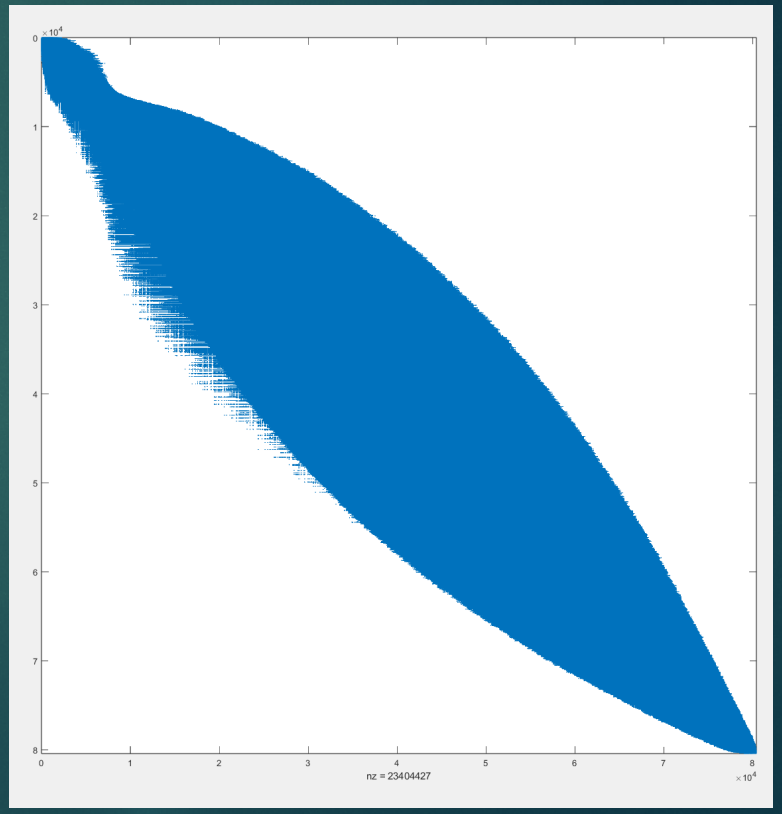
- Now that we have a discretized space to work with, we need to know how likely we are to move from one state to another.
- To determine this, we launch many trajectories from each point in the point cloud for a single period.
- The end points of these trajectories are tracked.
- However, not all the end points will lie in our discretized space.
- So, we simply identify the point in the point cloud closest to the end point of these trajectories and consider the trajectory to have ended there.

Transition Probability

- We now have data on where trajectories end for all the points in the point cloud.
- We can use this to form a transition probability from state i to state j.
- All these transition probabilities taken together form a transition probability matrix that fully describe the stochastic process.
- These matrices are really quite large.
  - ~50,000 x 50,000
- How big is this in memory?
- How do we store this in memory?!

Sparse Matrices

- The transition matrix turns out to be a very sparse matrix.
- That is, most of the elements are 0.
- This is an example of a transition matrix we work with.
- It's got about 4% of elements that are nonzero.
- Matlab has built in functionality to handle sparse matrices.
- For example, you can use spalloc to allocate the storage for a sparse matrix.
- Matlab will also speed up various operations involving sparse matrices.

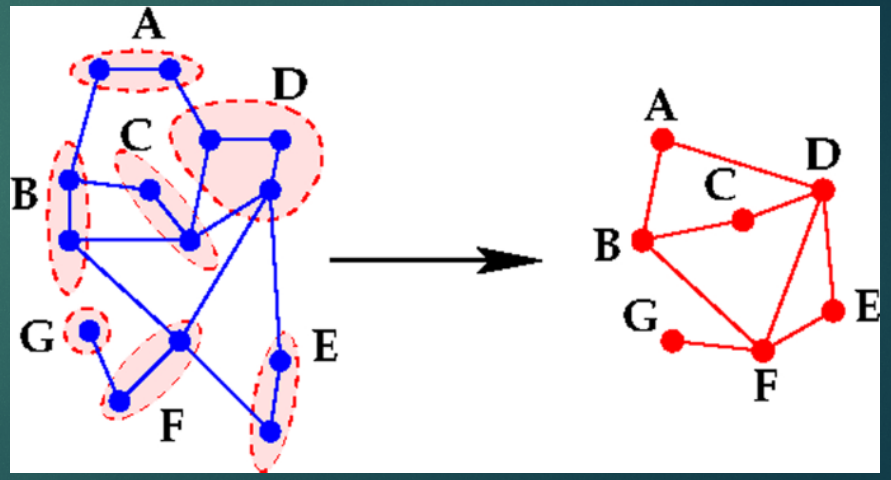


Strongly Connected Component

- We additionally only want a point cloud that is “Strongly connected”
- That is, we want a collection of points such that you can go from any point to any other point.
- You may have to pass through many other points, we do not have any disconnected components.
- This is a non-trivial problem.
- We use Matlab’s built in conncomp function to find the connected components.
- It uses a depth first search to look through all the connections between points.

Spatially Rarefy the States

- Returning to my point from before about spatially rarefying the states.
- We have a lot of states that are essentially sitting atop one another.
- We don't want that, it's a waste of compute power and memory.
- So, we coarsen the set!
- We look at the eigenvalues of the P matrix.
- This tells us which states can be used to represent the entire network efficiently.



Chen, J., Saad, Y. & Zhang, Z. Graph coarsening: from scientific computing to machine learning. *SeMA* **79**, 187–223 (2022). https://doi.org/10.1007/s40324-021-00282-x

How to determine the Markov Matrix

- To construct the Markov Transition Matrix, we follow a series of steps

1. Generate the Point Cloud
   - We generate a point cloud that samples the transition the pathway and the are around the attractors
2. From the point cloud, we simulate trajectories for one period.
   - We launch a large number of trajectories and track where they begin and end.
3. We identify which states in the point cloud are closest to the end of the trajectories and consider them to have ended there instead.
4. We form a transition probability matrix from this data.

Markov Chain Theorem and Probability Distributions

- With the transition matrix we can start to determine some interesting things about the matrix.
- If we start with a given state, we can use the transition matrix to determine the probability of being in a different state at the next time step.

$$x(t+1) = x(t) * P$$

- You can continue acting the matrix on the state vector to determine the probability at any point in time.

$$x(t) = x(0) * P^t$$

- The i^{th} vector component of $x(t)$ is the probability that the system will be in the i^{th} state at that time.

Steady State Probability Distribution

- We can determine the steady state probability distribution by finding the state vector that is unchanged when acted upon by the matrix P.
- That is the vector π such that,

$$\pi = \pi P$$

- How do we solve for π in this case?
- This is just an eigenvector problem.
- Google used to use a similar approach to rank their search results.
  - Google PageRank if you want some more details.

Transition Path Theory

- Transition path theory is an extension of transition state theory.
- Transition state theory is used to determine the rates of chemical reactions.
  - Or at least try to explain them.
- You may be familiar with the Arrhenius Equation.

$$k = Ae^{-\frac{E_a}{RT}}$$

Here, k is the rate constant, T is the temperature, A is the pre-exponential factor, E_a is the activation energy of the reaction, and R is the gas constant.

- Transition Path Theory is a more powerful formalism that allows us to talk more about the statistics of these transitions.

Transition Path Theory contd

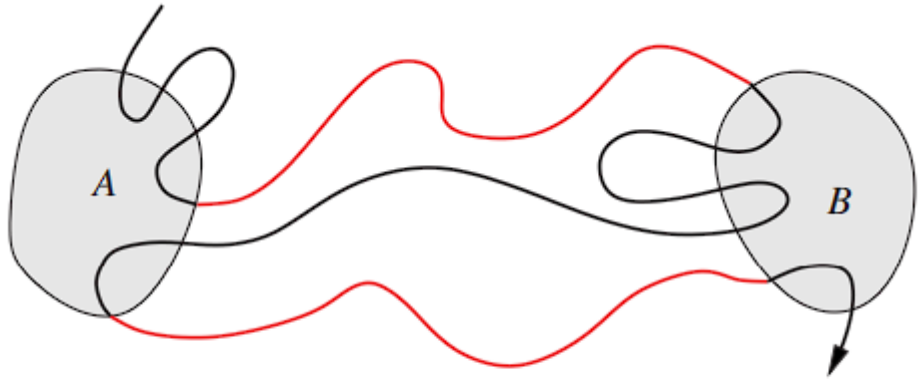


Fig. 1. Schematic representation of the reactant state A, the product state B, a piece of an equilibrium trajectory (shown in black) and the two reactive trajectories along it (shown in light gray)

Eric Vanden-Eijnden Transition Path Theory, Lect. Notes Phys. 703, 453–493 (2006)

- In transition state theory we talk about a state space S and two subsets of that space.
- A can be thought of as the reactants state.
- B can be thought of as the product state.

The Committor Function

- One of the most powerful aspects of the TPT formalism is the committor function.
- The committor function describes the transition process very broadly.
- There are technically two types of committor that we typically define:
  - $q^+(x)$: The forward committor is the probability that, if you start at the state x, $x(t)$ will proceed to B before returning to A.
  - $q^-(x)$: The backward committor is the probability that $x(t)$ was last at A instead of B.

Reactive Trajectories

- There are a lot of other parameters we can determine using the TPT formalism.
- Before we do, let's quickly define a "reactive reajectory"
- A reactive trajectory is a trajectory that is going from A to B without returning to A.

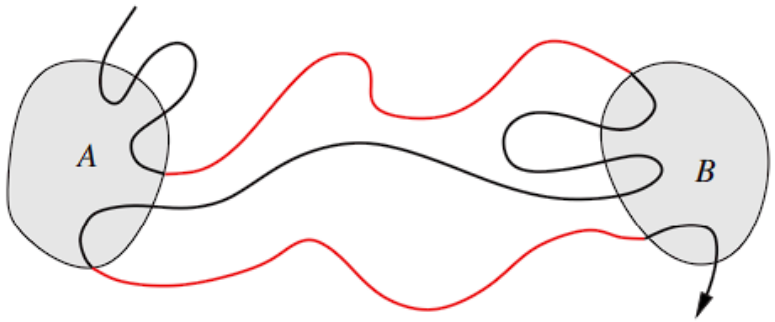


Fig. 1. Schematic representation of the reactant state A, the product state B, a piece of an equilibrium trajectory (shown in black) and the two reactive trajectories along it (shown in light gray)

Probability Density of Reactive Trajectories

- We can determine the probability that a state is going to be a part of a reactive trajectory pretty easily.

$$\mu_r(x) = q^+(x)q^-(x)\pi(x)$$

Here μ_r is the probability density of reactive trajectories, q^+ is the forward committor, $q^-$is the backward committor, and π is the steady state probability density.

- This equation should make intuitive sense.

Probability Current of Reactive Trajectories

- We can also define something called the probability current of reactive trajectories.
- This is the probability that the system will go from state i to state j while on a reactive trajectory.

$$f_{ij} = \pi_i q_i^- P_{ij} q_j^+$$

Here f_{ij} is the probability of going from state i to state j when on a reactive trajectory, q^+ is the forward committor, $q^-$is the backward committor, π_i is the probability that the system is in state i, and P_{ij} is the probability of going from state i to state j.

- This again should make some intuitive sense.

Transition Rate and Escape Rate

- We are very interested in determining when the system will transition.
- We capture this in the "transition rate"

$$\nu_{AB} = \sum_{i \in A} \sum_{j \in S} f_{ij} = \sum_{i \in S} \sum_{j \in B} f_{ij}$$

Here f_{ij} is the probability of going from state i to state j when on a reactive trajectory, and ν_{AB} is the transition rate.

- We also define the escape rate as

$$k_{AB} = \frac{1}{\rho_A} \nu_{AB}$$

Here ρ_A is the probability that we were last in A.

Mean First Passage Time

- One final parameter we are sometimes interested in is the mean first passage time or MFPT, μ_{ij}.
- The mean first passage time is how long it takes for the system to go from state i to state j for the first time.
- It can be calculated using

$$\mu_{ij} = 1 + \sum_{k \neq j} P_{ik}\mu_{kj}$$

$$\mu_{jj} = \frac{1}{\pi_j}$$

- Together, these form a system of equations you can solve to find the entire matrix, μ_{ij}.

One non-linear oscillator

- We've applied this approach to a single non-linear oscillator described by the following equation

$$\ddot{x} + 0.1\dot{x} + x + 0.3x^3 = 0.4\cos(1.4t) + \sqrt{0.05}\, dW$$

- This system has two attractors, one associated with a high amplitude oscillations and one with small oscillations.
- This system also has basins that are about equal in size.

Calculating the Committor

- With the transition matrix we can form the generator, L, of the stochastic process.
- With this generator, we can now compute the forward and backward committor functions.
- To do this, we solve the following system of equations:

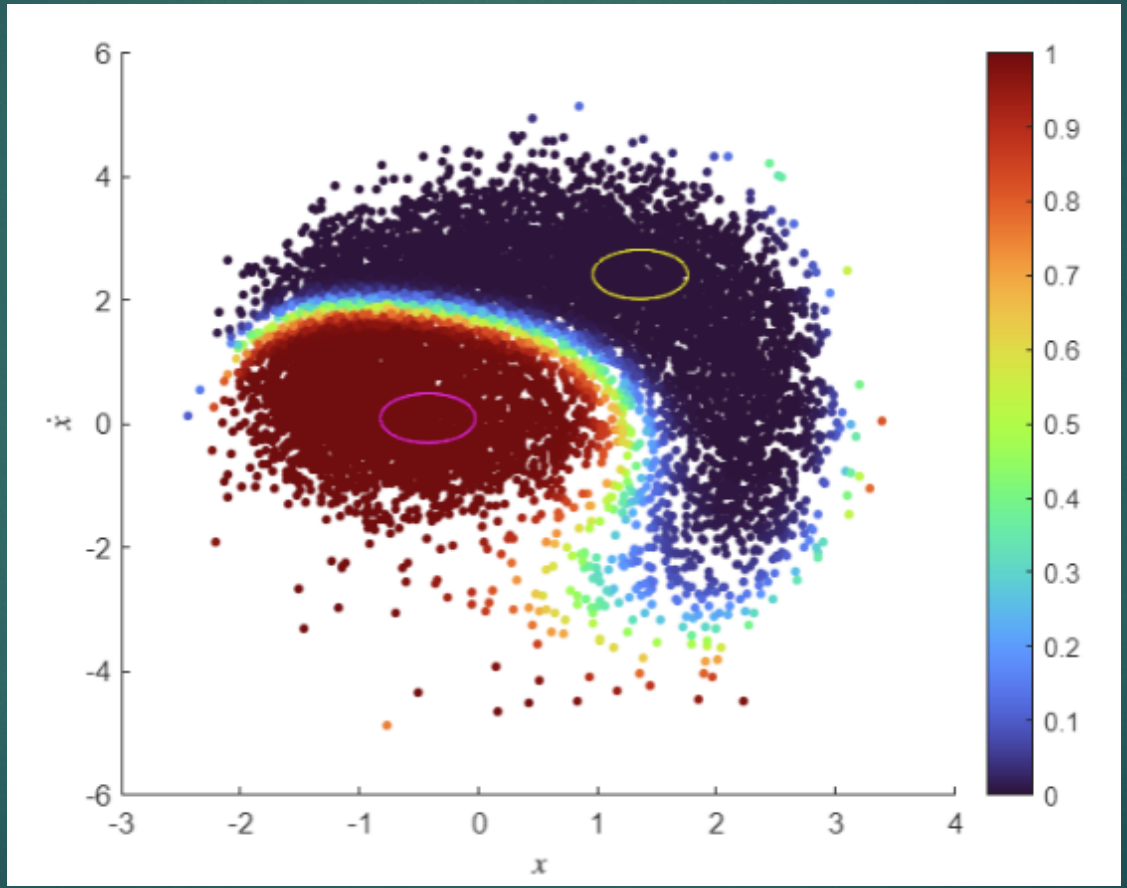
$$\sum_{k \in S} l_{ik} q_k^+ = 0, \quad \forall i \in (A \cup B)^C$$

$$q_i^+ = 0, \forall i \in A$$

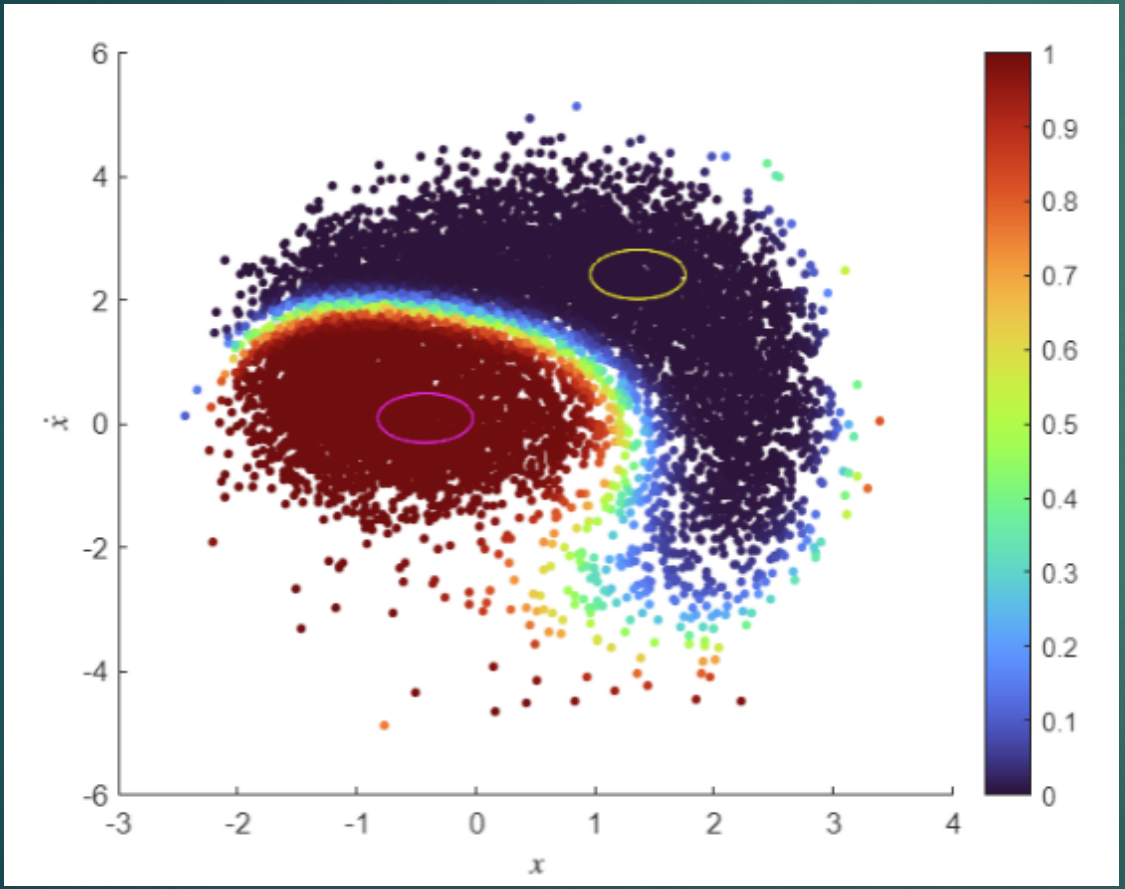
$$q_i^+ = 1, \forall i \in B$$

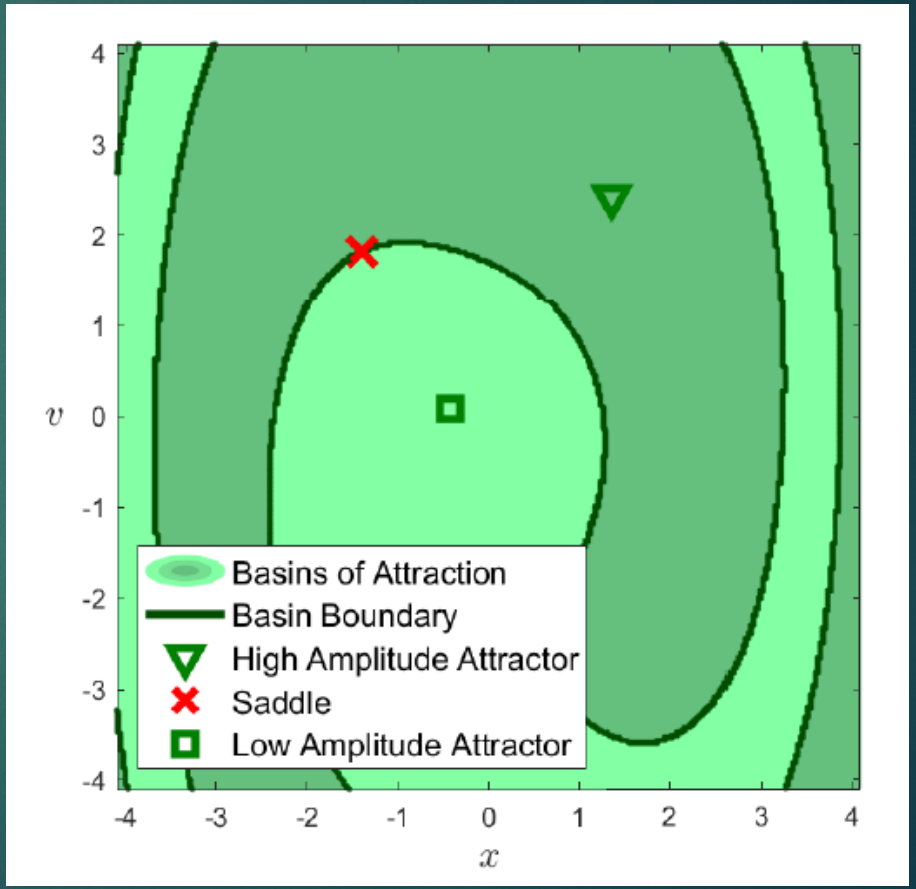
- Here l_{ik} is the ik^{th} element of L, A and B are the attractors, and S is the entire state space.

Committor Function



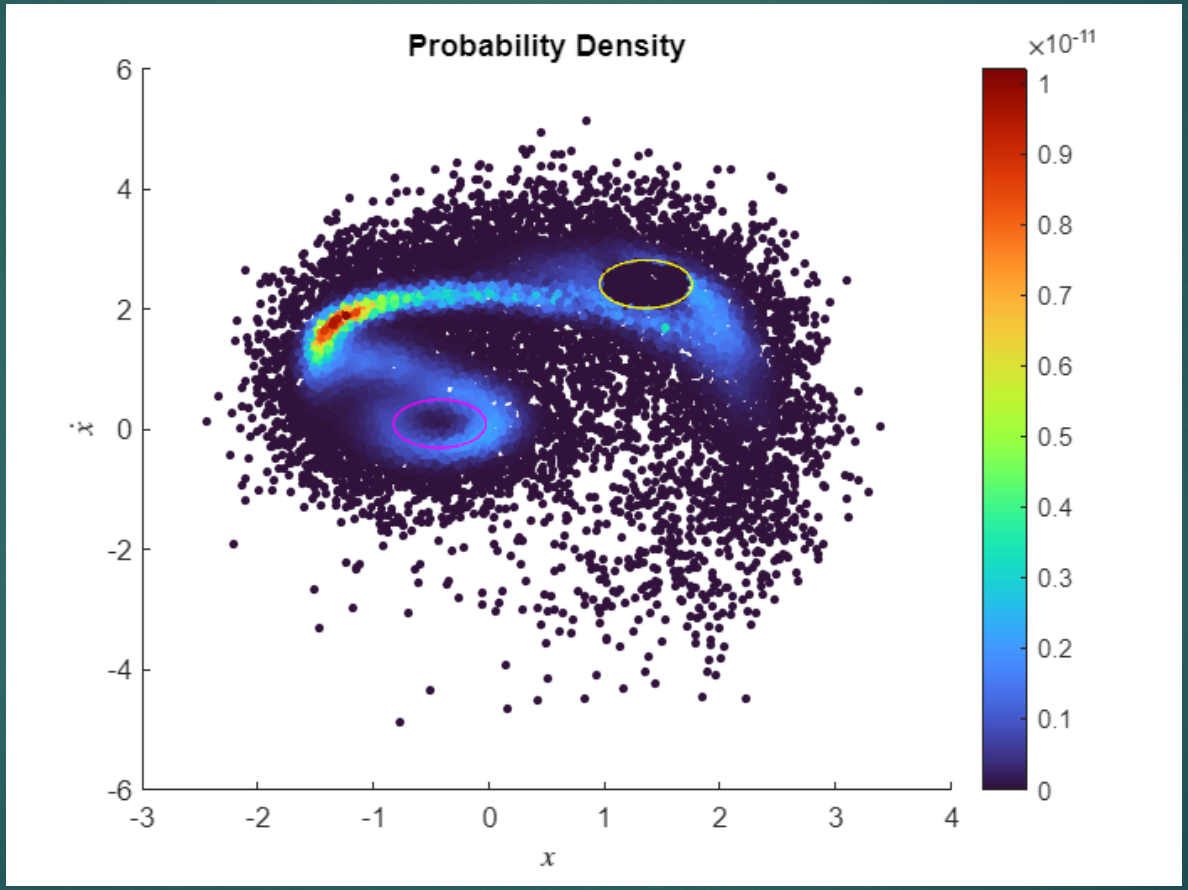
Analogue Markov Chain Committor comparison





Cilenti 2022

Probability Density Plot

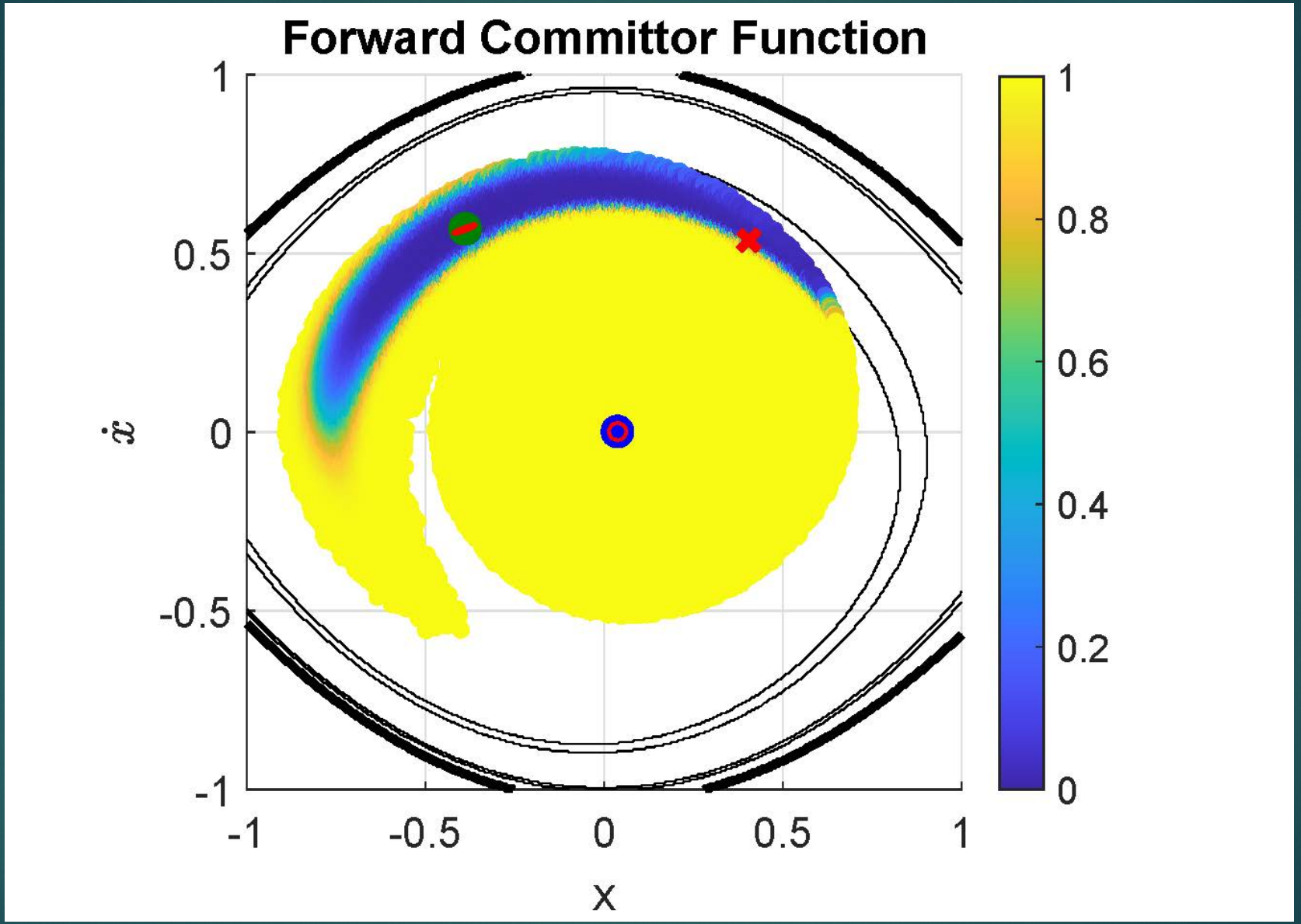


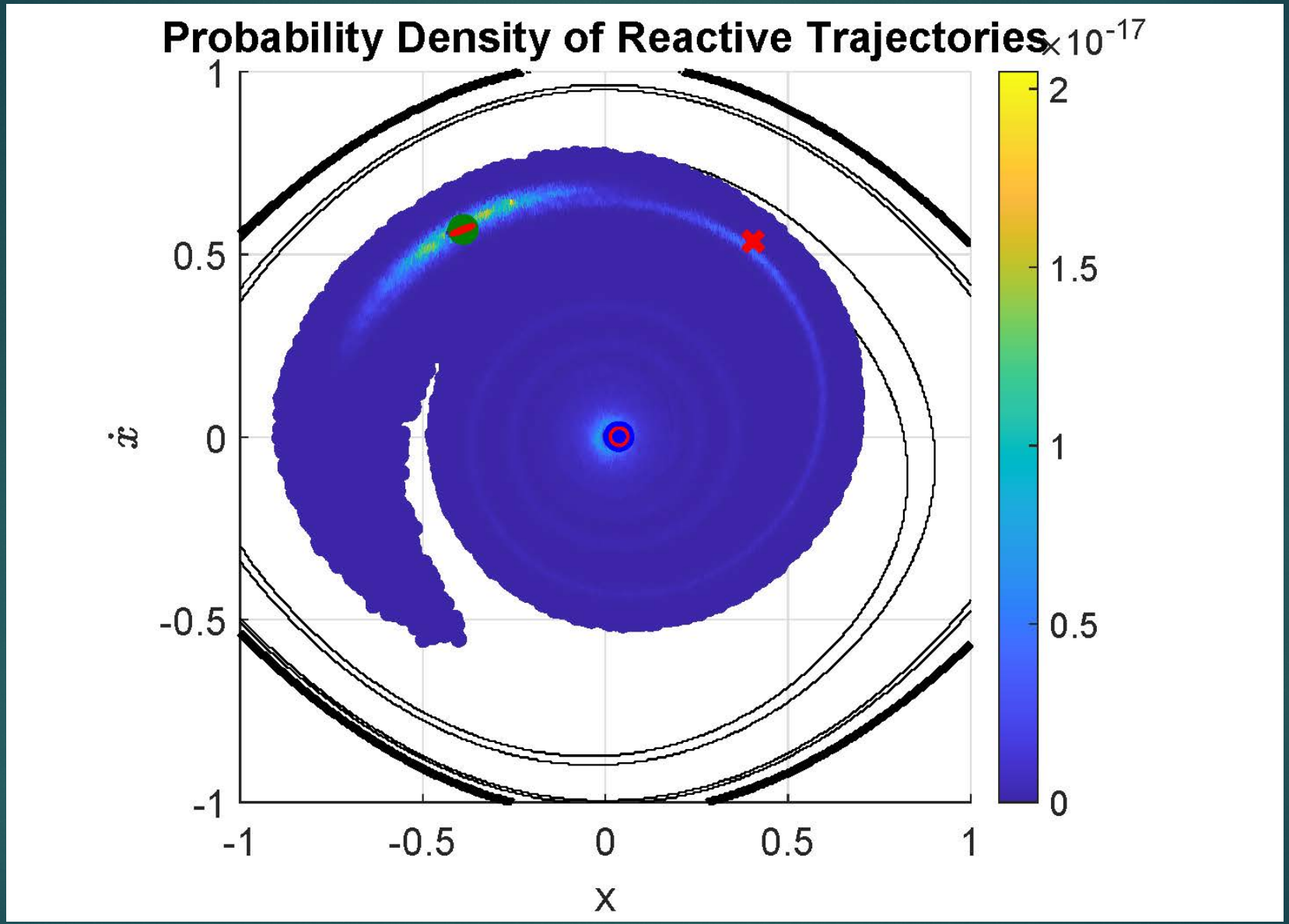
Application to Another System

- After the success with the first system, I also applied it to a system designed to replicate an experimental system Lautaro studied.
- Lautaro was able to determine values for the parameters to mimic his experimental set up.

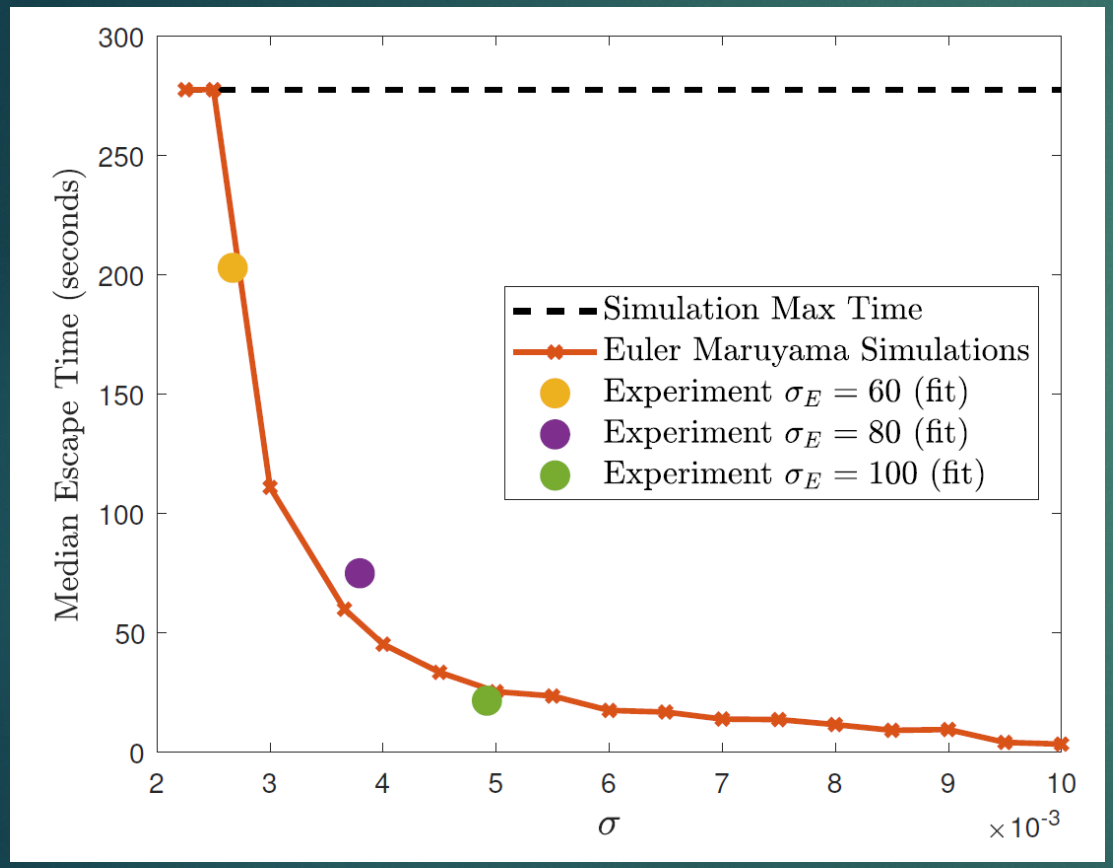
$$\ddot{x} + 0.009\dot{x} + x - 0.48x^3 = 0.007\cos(0.9t) + \sqrt{0.004}\, dW$$

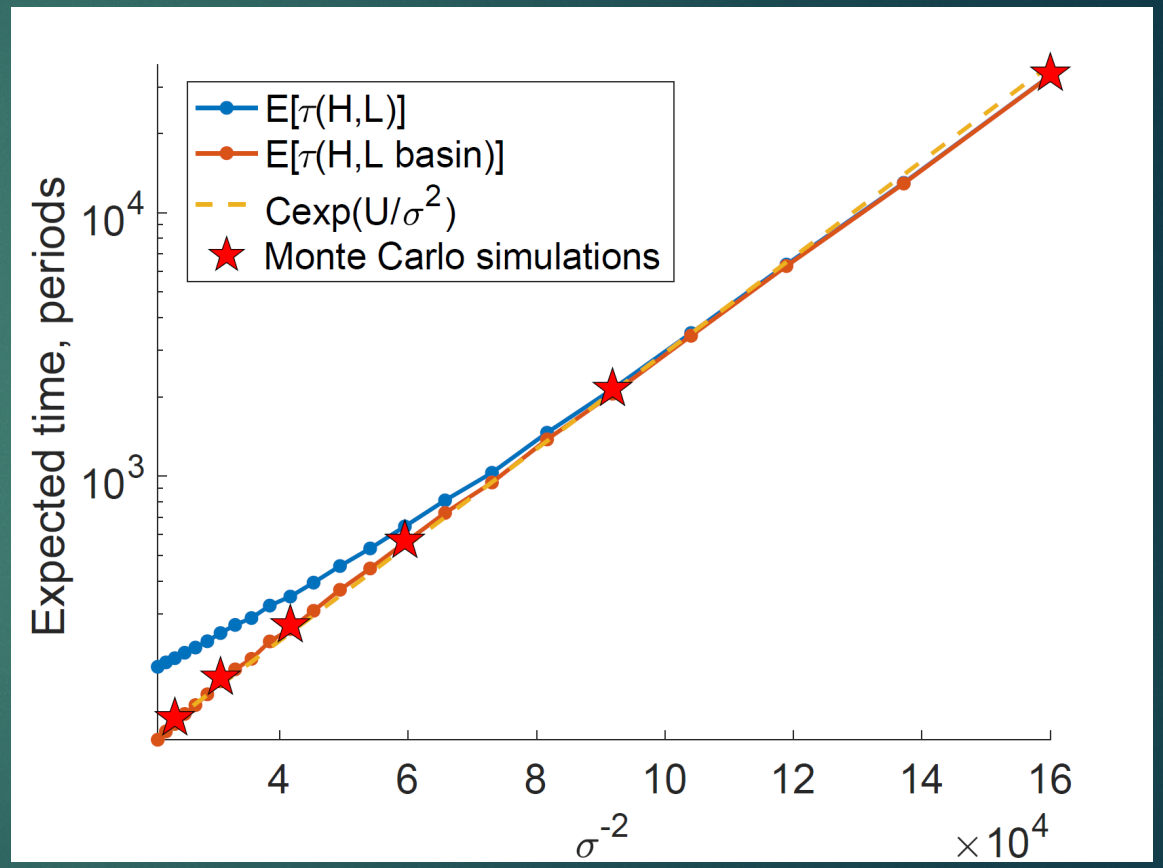
- This system is *not* as well behaved as the first system.


Forward Committor Function
1
0.5
0
-0.5
-1
$\dot{x}$
-1
-0.5
0
0.5
1
x
1
0.8
0.6
0.4
0.2
0


Probability Density of Reactive Trajectories
×10^-17
2
1.5
1
0.5
0
1
0.5
0
-0.5
-1
ẋ
-1
-0.5
0
0.5
1
x

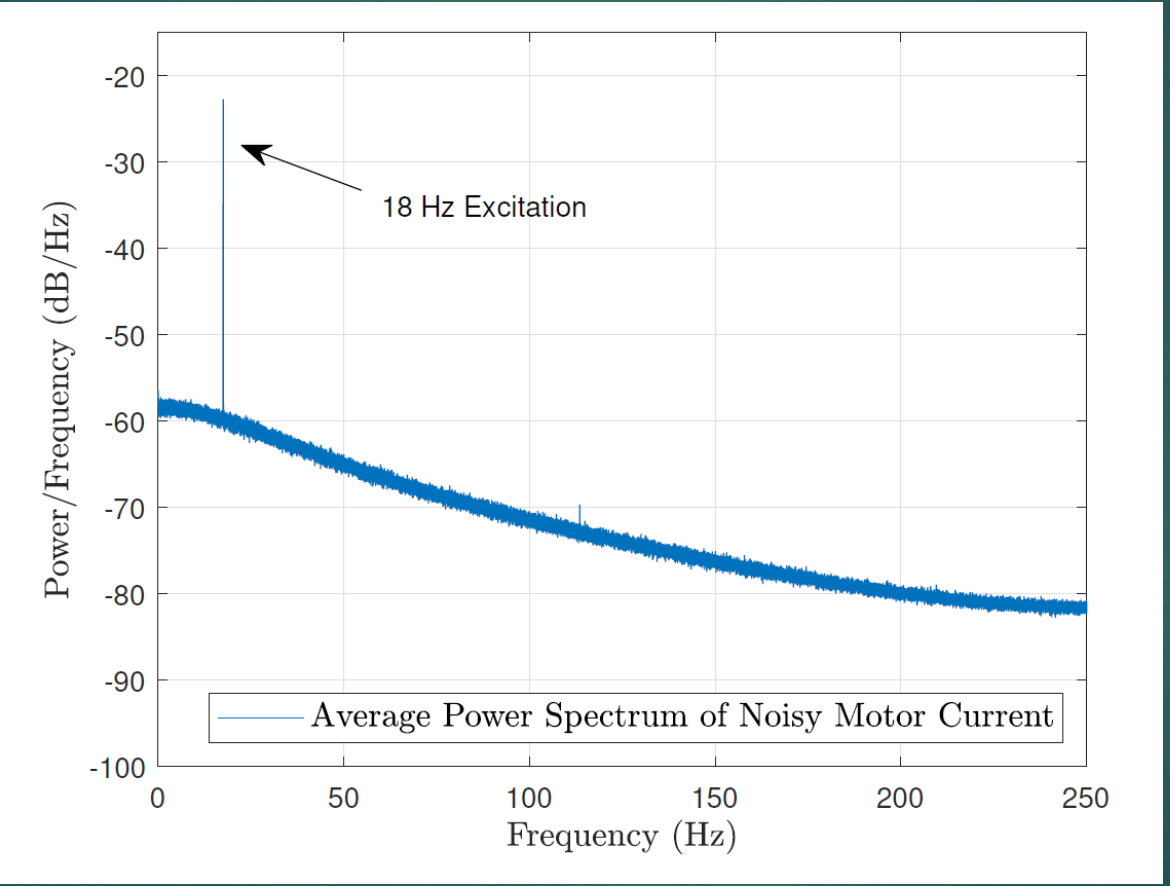
Escape Time vs Noise Level





Cilenti 2022

Experimental Noise

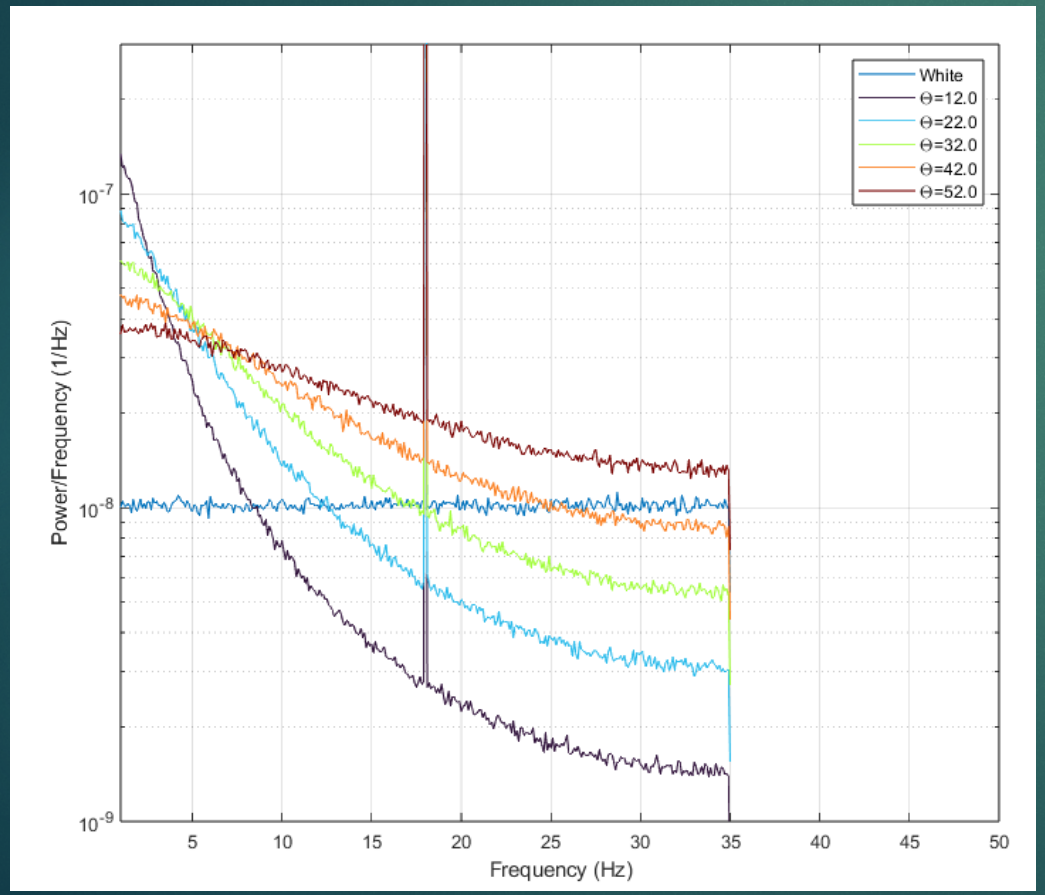


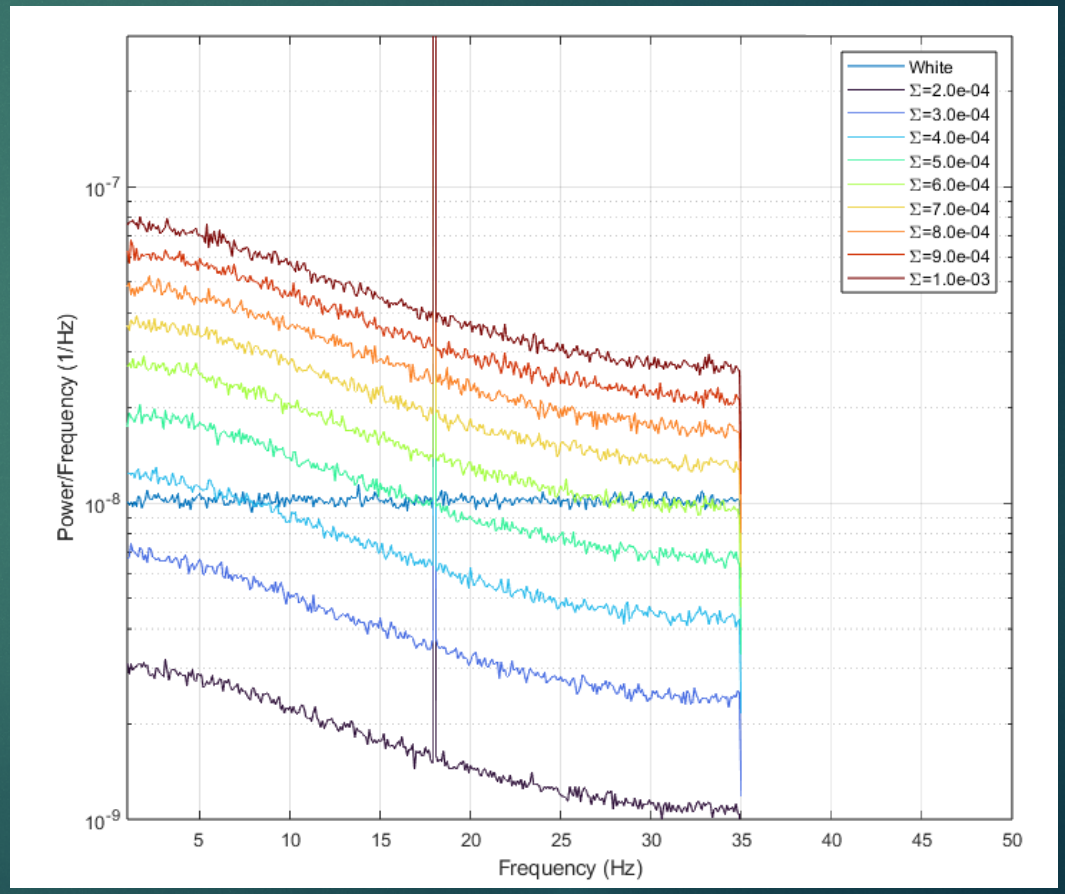
Cilenti 2022

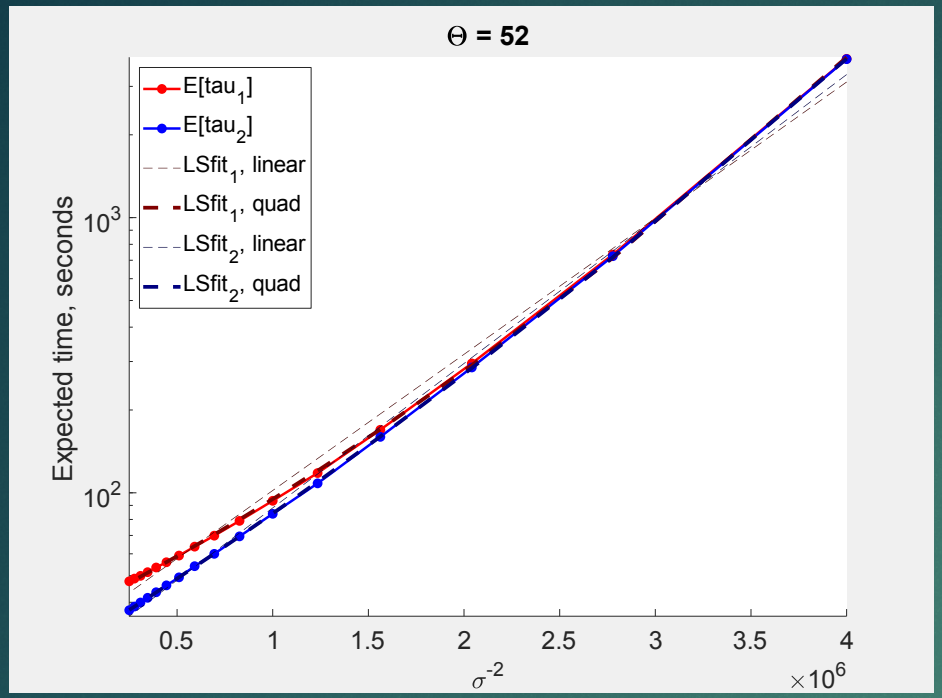
Numerical Colored Noise

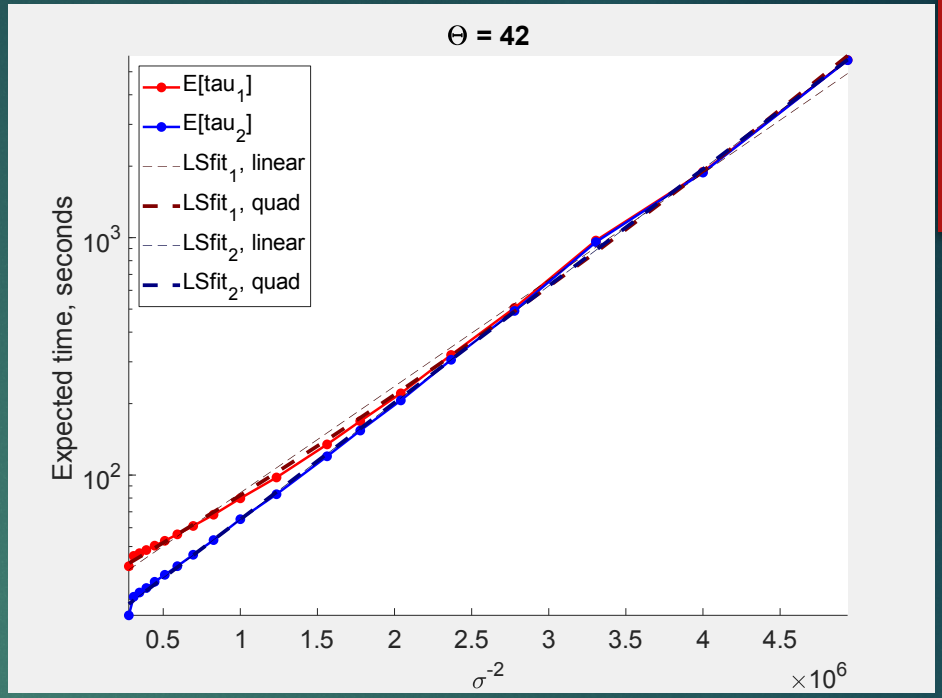
$$n(t+dt) = n(t) - \Theta\, dt\, n(t) + \Sigma\sqrt{2dt\Theta}\ \eta_i$$

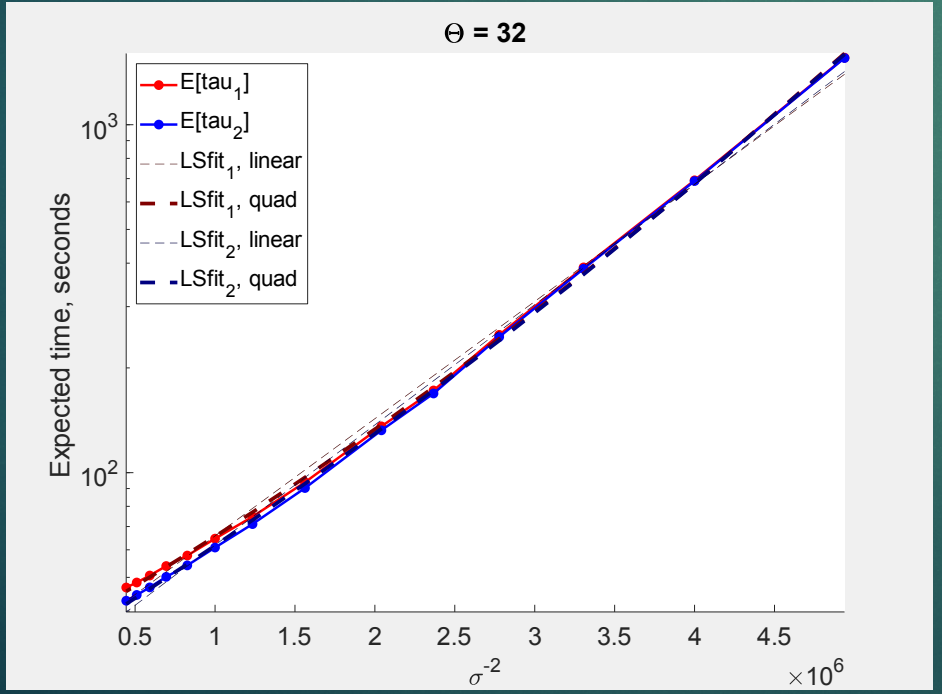
Θ: The correalation time Σ: The standard Deviation

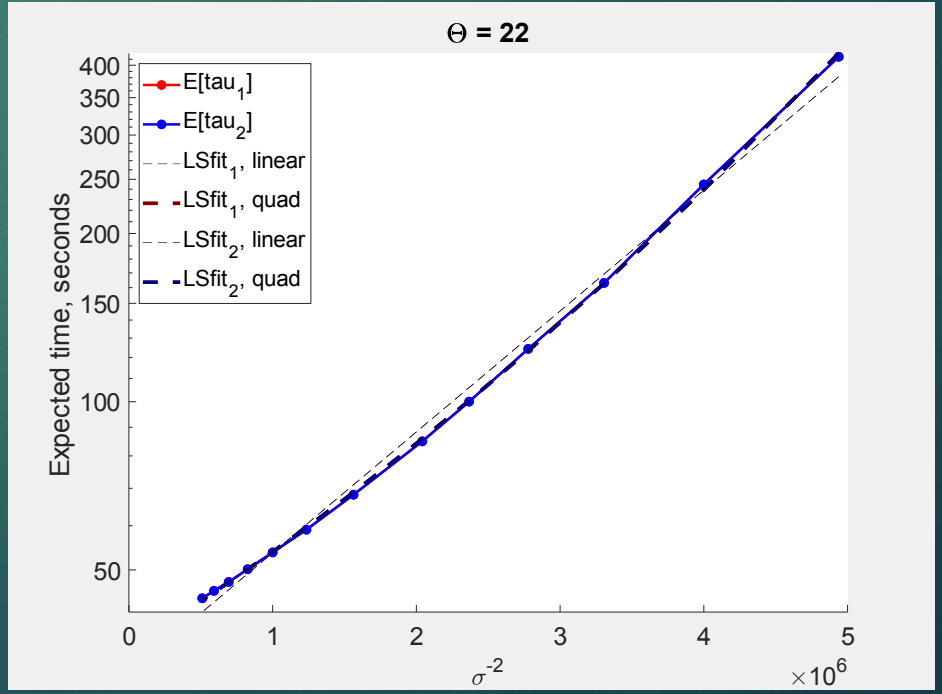





Θ = 52
E[tau1]
E[tau2]
LSfit1, linear
LSfit1, quad
LSfit2, linear
LSfit2, quad
Expected time, seconds
10^3
10^2
0.5
1
1.5
2
2.5
3
3.5
4
σ^-2
×10^6


Θ = 42
E[tau1]
E[tau2]
LSfit1, linear
LSfit1, quad
LSfit2, linear
LSfit2, quad
Expected time, seconds
10^3
10^2
0.5
1
1.5
2
2.5
3
3.5
4
4.5
σ^-2
×10^6


Θ = 32
E[tau1]
E[tau2]
LSfit1, linear
LSfit1, quad
LSfit2, linear
LSfit2, quad
Expected time, seconds
10^3
10^2
0.5
1
1.5
2
2.5
3
3.5
4
4.5
σ^-2
×10^6


Θ = 22
E[tau1]
E[tau2]
LSfit1, linear
LSfit1, quad
LSfit2, linear
LSfit2, quad
Expected time, seconds
400
350
300
250
200
150
100
50
0
1
2
3
4
5
σ^-2
×10^6

Generalizing to Coupled Oscillators

- One of the next steps is to extend this procedure to *coupled* oscillators.
- The equation that describes this system is given by:

$$\ddot{\boldsymbol{x}} + \delta\dot{\boldsymbol{x}} + \alpha\boldsymbol{x} + \beta\boldsymbol{x}^3 + \nu\boldsymbol{D}_N\boldsymbol{x} = \gamma\cos(\omega t) + \sqrt{\epsilon}\, dW$$

$$\boldsymbol{D}_N = \begin{bmatrix} 2 & -1 & & -1 \\ -1 & 2 & -1 & \\ \vdots & \ddots & \ddots & \ddots \\ -1 & & -1 & 2 \end{bmatrix}$$

- $\boldsymbol{x}$ now represents a vector of beam displacements, $\boldsymbol{D}_N$ is the coupling matrix, and ν is the coupling factor between oscillators.

Coupled Oscillators

- With coupled oscillators, there will be more basins and more transitions between the basins.
- We will need to effectively sample the state space to accurately compute the committor and its associated parameters.
- This will be more difficult not only because of the increased number of basins but also the increased dimensionality.
- With increasing dimensionality, we will need more points to get enough resolution of the space.

Future Directions

- We're studying the effect of the colored noise for a single oscillator., but there are a lot of interesting things we can explore with different colored noise.
- We also want to extend this to coupled oscillators.
- We have experimental data and results of MC simulations of these sorts of systems that we want to compare to.